

**What is Claimed:**

1. A method of using a first software module to invoke a second software module, the method comprising:

from the first software module, issuing a call to a first method that invokes a functionality performed by the second software module;

verifying that the call originated from a source that is permitted to invoke said functionality;

performing said functionality; and

returning to said first software module.

2. The method of claim 1, wherein said first method is performed by the second software module, said first method being exposed to the first module, said first method performing said functionality.

3. The method of claim 1, wherein said first method is performed by a third software module that invokes said functionality in the second software module.

4. The method of claim 3, wherein said call is made according to a first calling convention, and wherein said third software module uses a second calling convention different from said first calling convention to invoke said functionality in the second software module.

5. The method of claim 4, wherein said second calling convention causes a program stack to be modified in order to cause the next occurring return instruction to cause a return to the location to which a return would have occurred if a return had been executed following a call to said third software module and prior to a call to said second software module.

6. The method of claim 3, wherein said first calling convention comprises placing a first return address on a stack, said first return address representing a location at which execution is to resume after a function call is completed, and wherein said second calling convention comprises placing a second return address on a stack and placing data at the location represented by said

second return address, and wherein the method further comprises:

using said second return address to find said data, said data being used for at least one of:

performing said verifying act; and  
identifying a location to execute in order to perform said functionality.

7. The method of claim 1, wherein said verifying act comprises:

examining a call stack to identify a return address, and determining that the return address is part of a program module that is permitted, according to a standard or rule, to invoke said functionality.

8. The method of claim 7, further comprising:

determining that said return address is from a location or range of locations within said first program module from which invocation of said functionality is permitted to originate.

9. The method of claim 1, further comprising:

verifying that said first program module, or a portion thereof, has not been modified relative to a previously-known state.

10. The method of claim 1, wherein said first module is, or is part of, an application program.

11. The method of claim 1, wherein said second program module comprises a dynamic-link library.

12. A method of verifying a context in which a first program module has been called, the method comprising:

examining a call stack of a process in which said first program module executes to identify a return address to which control of the process will return upon completion of a call to said first program module;

determining that said return address is located within a second program module that

is permitted to call said first module; and

based on the result of said determining act, permitted execution of said first program module to proceed.

13. The method of claim 12, further comprising:

determining that said second program module, or a portion thereof, has not been modified relative to a previously-known state of said second program module, wherein said act of permitting execution of said second program module to proceed is further based on the determination as to whether said second program module has been modified.

14. The method of claim 12, wherein said first program module includes logic that resists misuse and/or tampering with said first program module, and wherein said determining act is performed by said first program module.

15. The method of claim 12, wherein said first program module comprises cryptographic functionality that stores and obscures a decryption key and that uses said decryption key to decrypt content, or uses said decryption key as part of a process of decrypting content.

16. The method of claim 12, wherein said first program module is called by a third program module, said third program module having a callable method exposed to said second program module, said callable method causing said first program module to be invoked and passing said return address to said first program module at the time that said first program module is invoked.

17. The method of claim 16, wherein said first program module adjusts the content of said call stack to reflect that said first program module will return to said return address upon completion of execution of said call to said first program module, said call stack reflecting, in the absence of the adjustment, that said first program module will return to an address other than said return address.

18. A program module comprising:

a function that is performable on behalf of a calling entity; and

logic that verifies an identity of the calling entity as a condition for performing said function, said logic consulting a call stack in order to identify said calling entity.

19. The program module of claim 18, wherein said logic determines said identity based on a return address on said call stack, said return address representing a location of an instruction to be executed when the program module completes execution.

20. The program module of claim 18, wherein said function is not exposed to said calling entity, and wherein said function is exposed to an intermediate entity that is callable by said calling entity, said intermediate entity calling upon the program module to perform said function on behalf of said calling entity.

21. The program module of claim 20, wherein said calling entity calls said intermediate entity using a first calling convention, and wherein said intermediate entity calls the program module using a second calling convention different from said first calling convention.

22. The program module of claim 21, wherein said calling entity calls said intermediate entity by calling a function in said intermediate entity and placing a first return address on said call stack, and wherein said intermediate entity calls the program module by calling or jumping to a location in said program module with one or more parameters including said first return address, wherein the program module verifies that said first return address is located within a calling entity that is allowed to call the program module, and wherein the program module adjusts said call stack so that the return address to be followed upon the next return instruction is equal to said first return address.

23. The program module of claim 21, wherein said first calling convention comprises placing said return address on said call stack, and wherein said second calling convention comprises placing a second return address on said call stack and placing data at the location represented by said second return address, said second return address being used to find said data, said data being

used to perform at least one of: verification of the identity of the calling entity; and identification of a location at which said function is located.

24. A computer-readable medium comprising computer-executable instructions to perform a method that facilitates verification of a call stack, the method comprising:

receiving a first call or first jump from a first entity, there being a call stack which, at the time of said first call or first jump, has a state in which the return address to be executed upon the next return instruction is equal to a first value;

issuing a second call or a second jump to a second entity, the second call or second jump being parameterized by one or more values including said first value, said second entity having access to said second value and using said second value to verify which return address was applicable at the time that said first call or said first jump was made, said second entity adjusting said call stack to set the return address equal to the first value.

25. The computer-readable medium of claim 24, wherein said second entity returns directly to said first entity without returning to an intermediate entity that received said first call or said first jump.